

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-185548

(43)Date of publication of application : 15.07.1997

(51)Int.Cl.

G06F 12/08

G06F 12/08

G06F 12/12

H03M 7/40

(21)Application number : 07-343170

(71)Applicant : SONY CORP

(22)Date of filing : 28.12.1995

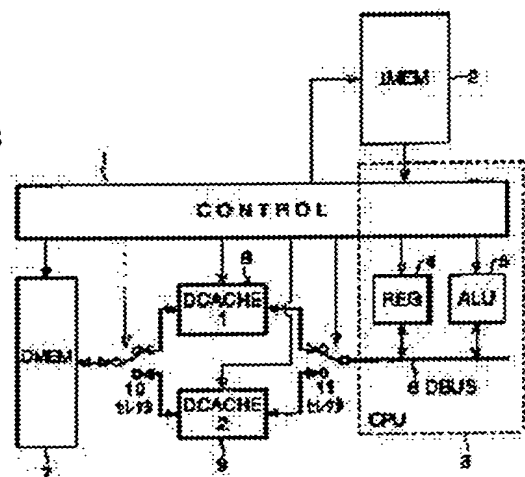
(72)Inventor : OKI MITSU HARU
TOTSUKA TAKUSHI

(54) CACHE MEMORY CONTROL METHOD, VARIABLE LENGTH ENCODING METHOD AND VARIABLE LENGTH DECODING METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To execute efficient encoding or decoding corresponding to code length.

SOLUTION: When prescribed input data is encoded, data to be referred to is stored in a data memory 7 and data corresponding to the code with comparatively short code length, which requires high speed reference, among data stored in the data memory 7 is transferred to a data cache 8 and is locked. ALU 5 constituting CPU 3 reads data stored in the data cache 8 through a data bus 6, a register file 4 and a selector 11 without a cache error, and encodes the input data based on the data.



LEGAL STATUS

[Date of request for examination]

13.09.2002

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the
examiner's decision of rejection or application
converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of
rejection][Date of requesting appeal against examiner's decision
of rejection]

[Date of extinction of right]

BEST AVAILABLE COPY

引用例 1 の写し

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-185548

(43) 公開日 平成9年(1997)7月15日

(51) Int.Cl. ^a	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 12/08	3 1 0	7623-5B	G 0 6 F 12/08	3 1 0 Z
		7623-5B		B
	12/12	7623-5B	12/12	D
H 0 3 M 7/40		9382-5K	H 0 3 M 7/40	

審査請求 未請求 請求項の数 6 O L (全 18 頁)

(21) 出願番号 特願平7-343170

(22) 出願日 平成7年(1995)12月28日

(71) 出願人 000002185

ソニー株式会社

東京都品川区北品川6丁目7番35号

(72) 発明者 大木 光晴

東京都品川区北品川6丁目7番35号 ソニー株式会社内

(72) 発明者 戸塚 卓志

東京都品川区北品川6丁目7番35号 ソニー株式会社内

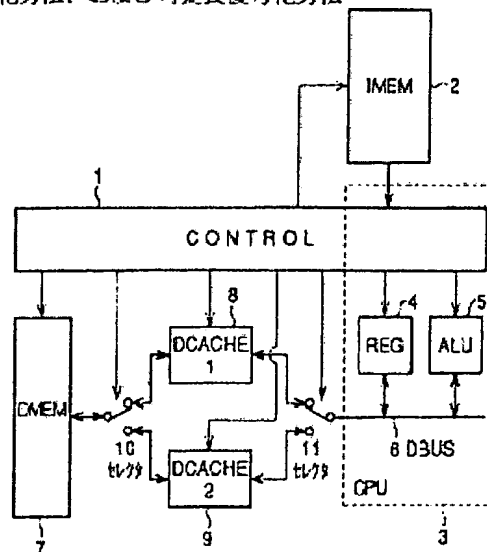
(74) 代理人 弁理士 稲本 義雄

(54) 【発明の名称】 キャッシュメモリ制御方法、可変長符号化方法、および可変長復号化方法

(57) 【要約】

【課題】 符号長に応じた効率的な符号化または復号化を可能にする。

【解決手段】 所定の入力データを符号化するとき参照するデータをデータメモリ7に記憶させ、データメモリ7に記憶されたデータのうち、高速な参照が必要な比較的短い符号長の符号に対応するデータをデータキャッシュ8に転送し、ロックする。CPU3を構成するALU5は、データ用バス6、レジスタファイル4、およびセレクタ11を介して、データキャッシュ8に記憶されたデータをキャッシュミスなしで高速に読み出し、それに基づいて、入力データを符号化する。



【特許請求の範囲】

【請求項 1】 データを第 1 のメモリに記憶し、前記データをキャッシュメモリに転送し、所定のプログラムを第 2 のメモリに記憶し、前記データを参照して、前記プログラムを実行する場合において、前記キャッシュメモリを制御するキャッシュメモリ制御方法において、前記データのうち、所定の時間内で参照すべきものを予め前記キャッシュメモリに転送し、前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記データの所定のことを前記キャッシュメモリに常駐させることを特徴とするキャッシュメモリ制御方法。

【請求項 2】 データを第 1 のメモリに記憶し、所定のプログラムを第 2 のメモリに記憶し、前記プログラムをキャッシュメモリに転送し、前記データを参照して、前記プログラムを実行する場合において、前記キャッシュメモリを制御するキャッシュメモリ制御方法において、前記プログラムのうち、所定の時間内で実行すべきものを予め前記キャッシュメモリに転送し、前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記プログラムの所定のことを前記キャッシュメモリに常駐させることを特徴とするキャッシュメモリ制御方法。

【請求項 3】 データを第 1 のメモリに記憶し、前記データをキャッシュメモリに転送し、所定のプログラムを第 2 のメモリに記憶し、前記データを参照して、前記プログラムを実行し、所定の入力データに対して可変長符号化を行い、符号化された圧縮データを生成する可変長符号化方法において、前記第 1 のメモリに、前記入力データに対応する符号化された前記圧縮データを記憶させ、前記第 1 のメモリに記憶された前記圧縮データのうち、前記圧縮データの符号長が所定のビット数より小さいものを、前記キャッシュメモリに予め転送し、前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記圧縮データを前記キャッシュメモリに常駐させることを特徴とする可変長符号化方法。

【請求項 4】 データを第 1 のメモリに記憶し、所定のプログラムを第 2 のメモリに記憶し、前記プログラムをキャッシュメモリに転送し、前記データを参照して、前記プログラムを実行し、所定の入力データに対して可変長符号化を行い、符号化された圧縮データを生成する可変長符号化方法において、前記入力データに対応する前記圧縮データを生成するプログラムを前記第 2 のメモリに記憶させ、前記第 2 のメモリに記憶された前記プログラムのうち、前記プログラムが生成する前記圧縮データの符号長が所定のビット数より小さいものを、前記キャッシュメモリに予め転送し、

前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記プログラムを前記キャッシュメモリに常駐させることを特徴とする可変長符号化方法。

【請求項 5】 データを第 1 のメモリに記憶し、前記データをキャッシュメモリに転送し、所定のプログラムを第 2 のメモリに記憶し、前記データを参照して、前記プログラムを実行し、所定の圧縮された入力データに対して可変長復号化を行い、前記入力データが復号化された復号化データを生成する可変長復号化方法において、前記第 1 のメモリに、圧縮された前記入力データに対応する前記復号化データを記憶させ、前記第 1 のメモリに記憶された前記復号化データのうち、対応する前記入力データの符号長が所定のビット数より小さいものを、前記キャッシュメモリに予め転送し、前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記復号化データを前記キャッシュメモリに常駐させることを特徴とする可変長復号化方法。

【請求項 6】 データを第 1 のメモリに記憶し、所定のプログラムを第 2 のメモリに記憶し、前記プログラムをキャッシュメモリに転送し、前記データを参照して、前記プログラムを実行し、所定の圧縮された入力データに対して可変長復号化を行い、前記入力データが復号化された復号化データを生成する可変長復号化方法において、前記入力データに対応する前記復号化データを生成するプログラムを前記第 2 のメモリに記憶させ、前記第 2 のメモリに記憶された前記プログラムのうち、前記プログラムが生成する前記復号化データに対応する前記入力データの符号長が所定のビット数より小さいものを、前記キャッシュメモリに予め転送し、前記キャッシュメモリをロックすることにより、前記キャッシュメモリに転送された前記プログラムを前記キャッシュメモリに常駐させることを特徴とする可変長復号化方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、キャッシュメモリ制御方法、可変長符号化方法、および可変長復号化方法に関し、例えば、画像データを符号化したり、符号化された画像データを復号化する場合に用いて好適なキャッシュメモリ制御方法、可変長符号化方法、および可変長復号化方法に関する。

【0002】

【従来の技術】 現在、画像圧縮の方式として、一番よく使われている方式が、MPEG 2 (Moving Picture Experts Group phase 2) である。MPEG 2 は、ISO (International Organization for Standardization:

国際標準化機構)が提案した画像圧縮方式であり、詳細は、文獻ISO/IEC13818-2に述べられている。

【0003】MPEG2の符号化方法においては、入力画像データに対して2次元8×8DCT(離散コサイン変換)をした後、量子化を行い、量子化されたデータをジグザグスキャンデータ列に並べ直し、VLC(バリエーション長コーディング)をして、圧縮を行っている。

【0004】量子化後のデータは2次元8×8行列上のデータとなるが、ジグザグスキャンデータ列とは、図5に示す矢印の順番にデータをスキャンして1列に並べることである。符号化方法のVLCでは、ジグザグスキャンデータ列をrunとlevelの2つより成るデータ(run, level)に変換し、この(run, level)のデータを符号化して圧縮している。ここで、runとは、ノンゼロ(有効データ)までのデータの個数、即ち、0(無効データ)がいくつ連続しているかをカウントした値であり、levelとは、runだけ0が続いた後のノンゼロの値である。

【0005】図5の場合、ジグザグスキャンデータ列は、"0, 0, 0, 1, 0, 18, 0, -2, 0, 0, ..., 0"となるので、0が3個の後、1、その後、0が1個、その後、18、その後、0が1個、その後、-2、そして残りは0となる。従って、図6に示すように、(run, level) = (3, 1)、(1, 18)、および(1, -2)となる。VLCでは、上記3個の(run, level)を順番に「(run, level)対(符号)」のテーブルを参照しながら、符号化していく。

【0006】図7乃至図10に、上記「(run, level)対(符号)」のテーブルを載せておく。このテーブルから分かるように、(run, level) = (3, 1)は"001110"と符号化される。テーブル上で(3, 1)に対応する符号は"00111s"と書かれているが、sとはlevelが正のときは0、負のときは1である。つまり、sとはlevelの符号である。

【0007】また、(run, level) = (1, 18)は、"000000000000100000"と符号化され、(run, level) = (1, -2)は、"0001101"と符号化される。従って、量子化後のデータが図5に示すような場合、図11に示すような符号"001110000000000000001000000001101"となる。

【0008】また、図7乃至図10に載っていない(run, level)に対しては、Escape符号の後に、図12に示す符号を割り当てる。Escape符号は図7に示すように、"000001"という符号である。この後、runに対応する符号を図12から選出し

て、この符号をEscape符号の後に付け加える。図12に示す符号は、runを6ビット2進数表示したときのビットそのものである。この後、levelに対応する符号を図12から選出して、この符号をrunの符号の後に付け加える。図12に示す符号は、levelを12ビット2の補数表示したときのビットそのものである。ちなみに、図7乃至図10に載っていない(run, level)に対しては、Escape符号も含めると24ビットの長さの符号が割り当てられる。

【0009】また、MPEG2において、(run, level)の一番最初がrun=0のときは特殊な符号が割り当てられており、さらに、(run, level)がこれ以上ないことを示すEndOfBlockという符号もあるが、本発明には関係がないので、ここではその説明は省略する。

【0010】以下、図13乃至図15を参照して、従来の情報処理装置の構成、およびこの装置により実行されるバリエーション長コーディング(VLC)について詳細に説明する。

【0011】図13は、従来の情報処理装置の一例の構成を示すブロック図である。図13に示した装置は、一般的な情報処理装置である。

【0012】図13において、制御回路(CONTROL)1は、装置全体を制御するようになされている。インストラクションメモリ(MEM)2は、所定のインストラクション(プログラム)のコード等を記憶する。データメモリ(DMEM)7は、図7乃至図10、および図12に示したテーブルのデータ等を記憶する。レジスタファイル(REG)4は、制御回路1からの指令に基づいて、後述するセクタ10、データキャッシュ(DCACHE)1またはデータキャッシュ2、セクタ11、およびデータ用バス(DBUS)5を介してデータメモリ7より供給されたデータを記憶し、算術演算回路(ALU)5に供給するようになされている。データ用バス6は、レジスタファイル4と算術演算回路5との間で、データの受渡しを行うようになされている。

【0013】レジスタファイル4、算術演算回路5、データ用バス6、および制御回路1の一部(点線で囲まれた部分)は、いわゆるCPU(Central Processing Unit)3を構成している。

【0014】セクタ10は、制御回路1によって制御され、データメモリ7からのデータをデータキャッシュ8またはデータキャッシュ9のいずれかに供給するようになされている。データキャッシュ8およびデータキャッシュ9は高速にデータの読み書きが可能なメモリであり、データキャッシュ8は、セクタ10を介してデータメモリ7からデータが供給されたとき、制御回路1の制御により、データメモリ7からのデータを記憶する。また、データキャッシュ9は、セクタ10を介してデータメモリ7からデータが供給されたとき、制御回路1

の制御により、そのデータを記憶するようになされている。

【0015】セレクト11は、制御回路1の制御に基づいて、データキャッシュ8またはデータキャッシュ9のいずれかに記憶されたデータをデータバス6を介して、レジスタ4に供給するようになされている。

【0016】インストラクションメモリ2には、処理すべきプログラムの内容（コード）が格納されている。制御回路1からの制御により、インストラクションメモリ2内のプログラムのコードが順番に読み出され、読み出されたコードを制御回路1が解釈して、適切なデータをデータメモリ7から読み出し、レジスタファイル4と算術演算回路5を使用して所定の演算を行い、データメモリ7に演算結果を戻すようになされている。

【0017】上記処理すべきプログラムの内容とは、図14のフローチャートを参照して後述するように、例えば、ステップS3、S4、およびS5に対応するプログラムであり、データメモリ7上にあるデータ（run, level）をレジスタファイル4にロードしたり、runの値が31を越えているか否か、あるいは、levelの絶対値が40を越えているか否かを算術演算回路5に判定させたり、あるいは、算術演算回路5に（levelの絶対値-1）×32+runを計算させ、その計算結果を変数ptrの値とし、それをデータメモリ7において変数ptrというデータが格納されている番地にストアさせたりすることである。

【0018】また、図14のステップS6に対応するプログラムとして、データメモリ7上にある、a[0]乃至a[32×40-1]、およびb[0]乃至b[32×40-1]の内、a[ptr]、およびb[ptr]のデータをレジスタファイル4にロードすることである。さらにまた、ステップS7に対応するプログラムとして、上記レジスタファイル4にロードされたデータb[ptr]の値が、24であるか否かを算術演算回路5に判定させることである。

【0019】次に、図14のフローチャートを参照して、図13に示した情報処理装置の動作について説明する。

【0020】最初、ステップS1において、初期設定が行われる。即ち、図15に示したフローチャートで表される初期設定のサブルーチンが実行される。図15のステップS21においては、図7乃至図10に示したテーブルのデータ（符号）を格納しておく領域a[0]乃至a[32×40-1]と、その符号がどれほどの長さ（ビット長）であるかを記憶しておくための領域b[0]乃至b[32×40-1]がデータメモリ7上に確保される。

【0021】そして、図7乃至図10に示す（run, levelの絶対値）に対応する各符号が、a[(levelの絶対値-1)×32+run]に上位側につ

てそれぞれ格納され、それらの符号の長さ（ビット長）が、b[(levelの絶対値-1)×32+run]にそれぞれ格納される。対応する（run, levelの絶対値）が図7乃至図10に示したテーブルにないために、またデータ（符号）が格納されていない残りのb[j]（ここで、jは、まだ符号が格納されていない領域を示す）には、符号の最大のビット長である値24が格納される。

【0022】但し、図7乃至図10に示す符号の最後のsについては無視される。つまり、例えば、（run, levelの絶対値）=（1, 2）の符号は"000110s"であるから、sは無視され、e[(levelの絶対値-1)×32+run]=e[33]="000110"（バイナリデータ、上位ビット側につめられる）とされ、b[(levelの絶対値-1)×32+run]=b[33]=6とされる。

【0023】また、図7乃至図10には、runが32以上のものはない。また、levelの絶対値が41以上のものもないので、（run, levelの絶対値）が決まれば、（levelの絶対値-1）×32+runは一意に決まる。また、異なる2つの（run, levelの絶対値）においては、（levelの絶対値-1）×32+runの値が同一となることはない。さらに、図7乃至図10に示されている符号は全て23ビット以下であり、b[(levelの絶対値-1)×32+run]は23以下である。

【0024】ステップS21における初期化処理が終了すると、リターンし、図14のステップS2に進む。

【0025】ステップS2においては、制御回路1により、読み込むべき（run, level）があるか否かが判定される。読み込むべき（run, level）があると判定された場合、ステップS3に進み、（run, level）が読み込まれる。

【0026】次に、ステップS4において、制御回路1に制御された算術演算回路5により、読み込まれたrunの値が31を越えているか、または読み込まれたlevelの絶対値が40を越えているか否かが判定される。runの値が31を越えているか、またはlevelの絶対値が40を越えている場合、ステップS9に進む。図7乃至図10に示したテーブルには、runが31を越えるものはない。また、levelの絶対値が40を越えるものもない。従って、このステップS4において、runの値が31を越えているか、またはlevelの値が40を越えていると判定された場合、ステップS9において、図12に示した符号が割り当てられることになる。

【0027】一方、runの値が31を越えておらず、かつlevelの絶対値が40を越えていない場合、ステップS5に進み、変数ptrに（levelの絶対値-1）×32+runが代入される。そして、ステップ

S6において、変数`codetmp`に`a[ptr]`が、また、変数`code length`に`b[ptr]`が代入される。

【0028】次に、ステップS7に進み、制御回路1に制御された算術演算回路5により、`code length`の値が24であるか否かが判定される。`code length`の値が24であると判定された場合、ステップS9に進む。

【0029】`run`の値が31以下であり、かつ、`level`の絶対値が40以下であっても、図7乃至図10に示したテーブルに載っていないものがある。ステップS1のサブルーチン（図15）内のステップS21における初期設定処理において、図7乃至図10に示したテーブルに載っている $(run, level)$ に対応する $b[k]$ （ここで、 $k = (levelの絶対値 - 1) \times 32 + run$ ）は全て23以下であり、テーブルに載っていない $(run, level)$ に対応する $b[j]$ （ここで、 j は k 以外で可能な整数）は全て24であるように設定されている。従って、この判定処理で、 $b[j]$ が24であると判定された場合は、図12に示す符号を割り当てる必要がある。

【0030】ステップS9においては、まず、`Escape`符号である“000001”が出力され、次に、図12に示したように、`run`の値を6ビット2進数で表現したビット列が出力され、さらに、`level`の値を12ビット2の補数で表現したビット列が出力される。その後、ステップS2に戻り、ステップS2以降の処理が繰り返し実行される。

【0031】一方、`code length`の値が24でないと判定された場合、すなわち、ステップS3において読み込まれた $(run, level)$ に対応する符号が、図7乃至図10に示したテーブルに載っている場合のみ、ステップS8の処理が実行される。ステップS8においては、`codetmp`の上位ビット側から`code length`ビット分だけ出力される。 $a[(levelの絶対値 - 1) \times 32 + run]$ の上位ビット側から $b[(levelの絶対値 - 1) \times 32 + run]$ の長さ分には、図7乃至図10のテーブルに示した符号が格納されているので、これに基づいて、ステップS3で読み込まれた $(run, level)$ に対応する符号が出力される。

【0032】但し、図7乃至図10のテーブルに示された符号の最後の`s`に対応する信号がないので、`level`の値が正であれば0を、負であれば1を最後に出力する。例えば、 $(run, level) = (1, -2)$ がステップS3で読み込まれたときは、 $a[(levelの絶対値 - 1) \times 32 + run] = a[33] = 000110$ （バイナリデータ）と $b[(levelの絶対値 - 1) \times 32 + run] = b[33] = 6$ が、ステップS6において`codetmp`と`code length`に

それぞれ代入されるので、ステップS8において、`codetmp`の上位6ビットである“000110”が読み出され、そして、その後、`level`の符号である1が付け加えられて、出力される。

【0033】ステップS8の処理が終了した後、ステップS2に戻り、次の $(run, level)$ の処理に移る。

【0034】図5、および図6に示した例では、最初にステップS3において $(run, level) = (3, 1)$ が読み込まれ、ステップS4乃至S8の処理が行われ、次にステップS3において $(run, level) = (1, 18)$ が読み込まれ、ステップS4乃至S8の処理が行われ、さらにステップS3において $(run, level) = (1, -2)$ が読み込まれ、ステップS4乃至S8の処理が行われる。そして、図11に示す符号が出力される。

【0035】このようにして、VLCが実行される。

【0036】図13に示した情報処理装置には、上述したようにデータキャッシュ（DCACHE）8、およびデータキャッシュ（DCACHE）9があり、データメモリ7が直接、データ用バス6につながっていない。つまり、データメモリ7は、データキャッシュ8あるいはデータキャッシュ9を介してデータ用バス6につながっている。

【0037】データメモリ7は大容量のメモリであり、アクセスするのに時間がかかり過ぎる。そこで、データメモリ7の一部をデータキャッシュ8あるいはデータキャッシュ9にコピーする。データキャッシュ8とデータキャッシュ9は小容量のメモリであり、アクセス時間は短い。従って、使用するデータがデータキャッシュ8あるいはデータキャッシュ9にコピーされていれば、このデータをデータキャッシュ8あるいはデータキャッシュ9から高速に読み出し、データ用バス6を介してレジスタファイル4にロードすることができる。

【0038】このように、データキャッシュ8またはデータキャッシュ9を使うことにより、データを高速に読み出して、レジスタファイル4にロードすることができる。但し、データキャッシュ8およびデータキャッシュ9は小容量であり、データメモリ7の全てのデータをコピーすることは出来ない。もし、ロードする際に、データキャッシュ8あるいはデータキャッシュ9にデータのコピーが見つからなければ、これをキャッシュミスと呼び、データメモリ7からデータを読み出すと共に、データキャッシュ8あるいはデータキャッシュ9にそのコピーを置く操作を行なう。この間、図13に示した装置は計算を一時中断する。そして、コピーし終わった時点で計算を再開する。

【0039】セレクトタ10は、データキャッシュ8およびデータキャッシュ9のどちらをデータメモリ7にアクセス（接続）させるかをセレクトするようになされてい

る。セクタ11は、データキャッシュ8およびデータキャッシュ9のどちらをデータ用バス6にアクセス（接続）させるかをセレクトするようになされている。

【0040】データキャッシュは、最近では1つではなく2つ（この実施例の場合、データキャッシュ8およびデータキャッシュ9）に別れている。即ち、2-way-set-associativeの構成をとっている。データメモリ7におけるある特定の番地のデータは、データキャッシュ8の特定の番地にしかコピーされない。同様に、データメモリ7におけるある特定の番地のデータは、データキャッシュ9の特定の番地にしかコピーされない。つまり、データメモリ7におけるある特定の番地（ADRS）のデータは、データキャッシュ8もしくはデータキャッシュ9の特定の番地（CADRS）にしかコピーされない。

【0041】通常、 $CADRS = (ADRS) \bmod Cs$ なる式を用いて、データキャッシュ上の番地（CADRS）を算出する。ここでCsはキャッシュ（データキャッシュ8とデータキャッシュ9）の総バイト数を2で割ったものである。したがって、データメモリ7のある特定の番地（ADRS）に対応するデータのコピーを、データキャッシュ8またはデータキャッシュ9から読み出すには、データキャッシュ8もしくはデータキャッシュ9の所定の番地（CADRS）の箇所（キャッシュエントリと呼ぶ）に、データメモリ7の番地（ADRS）に対応するデータのコピーが存在するかどうかをチェックすればよい。そして、データキャッシュ8またはデータキャッシュ9のいずれにもこのデータのコピーが存在しなければキャッシュミスとなる。

【0042】このように2-way-set-associativeでは2箇所のチェックで済み、fully-associativeのようにデータキャッシュの全ての番地をチェックする必要はない。これにより、このチェックをすばやく行うことができる。

【0043】2-way-set-associativeの構成では、データメモリ7内の特定の3つ以上のデータが、データキャッシュ8の1つの共通の番地にしかコピーされないことがある。同様に、データメモリ7内の特定の3つ以上のデータが、データキャッシュ9の1つの共通の番地にしかコピーされないことがある。つまり、データメモリ7上の3個以上の異なる番地（A1、A2、A3、...）に対して、同一の番地（CADRS）をもつキャッシュエントリがデータキャッシュ8およびデータキャッシュ9に割り当てられることがある。

【0044】一般に、 $An = ADRS + n \times Cs$ （nは整数）で与えられる番地（An）に対して、番地（ADRS）と同一のキャッシュエントリが割り当てられる。ところが、同一の番地（CADRS）をもつキャッシュエントリはデータキャッシュ8に1個、データキャッシ

ュ9に1個の合計2個しかないので、3つのデータを順次、ロードしたいときには、1つ目のデータはデータキャッシュ8にコピーされていて、2つ目のデータはデータキャッシュ9にコピーされていて、キャッシュミスなしに1つ目と2つ目のデータをロードできる可能性があるが、3つ目のデータは必ずキャッシュミスを起こす。なぜなら、3つ目のデータがコピーされるべき番地は、既に1つ目と2つ目のデータによってそれぞれ占有されており、3つ目のデータはデータキャッシュ8とデータキャッシュ9には絶対にコピーされないからである。

【0045】このようなときには、3つ目のデータをデータキャッシュ8の1つ目のデータの上、あるいはデータキャッシュ9の2つ目のデータの上にオーバーライトすることによりコピーする。そして、データキャッシュ8あるいはデータキャッシュ9にコピーされた3つ目のデータをロードする。このように、コピーしようとする番地が既に他のデータにより占有されている場合、キャッシュミスを起こすが、このとき、データキャッシュ8またはデータキャッシュ9のいずれにオーバーライトするかは、制御回路1において、自動的に決定される。このとき、制御回路1は、Random方式やLRU（least-recently used）方式等の決定方法に従って、この決定を行うことができる。

【0046】キャッシュ、2-way-set-associative、fully-associative、Random方式、およびLRU方式については、一般的に知られているので、以上で説明を終わり、その詳細な説明は省略する。詳細については、例えば、Morgan Kaufmann Publishers, Inc. より発刊されている「COMPUTER ARCHITECTURE A QUANTITATIVE APPROACH（著者：JOHN LHMNESY, DAVID A PATTERSON）」などを参照のこと。

【0047】また、最近では、キャッシュミスを起こした場合、データキャッシュ8あるいはデータキャッシュ9のどちらにコピーするかを、インストラクションメモリ2のプログラムの中に明示しておくことができるようになった。もちろん、明示しなければ、制御回路1により、Random方式やLRU方式などを使って自動的に決定されることになる。これは、プログラムを書くプログラマが、より細かくキャッシュを操作できるようにとの配慮からである。このような機構を持ったキャッシュの制御を、ロックابل（lockable）なキャッシュ制御と呼ぶ。

【0048】例えば、インストラクションメモリ2上にデータキャッシュ9をロックする命令を書いておく。そして、データキャッシュ8およびデータキャッシュ9上にはないデータをレジスタファイル4にロードしようとすると、キャッシュミスが起き、データメモリ7からその

データをデータキャッシュ8あるいはデータキャッシュ9にコピーする。通常は、制御回路1により、データキャッシュ8またはデータキャッシュ9のいずれにデータをコピーするかが決定されるが、この場合は、データキャッシュ9がロックされているので、データキャッシュ9へのコピーは禁止され、データキャッシュ8にコピーされる。そして、データキャッシュ8にコピーされたデータがレジスタファイル4にロードされる。

【0049】

【発明が解決しようとする課題】例えば、図6に示すデータ(run, level)の例では、図14のステップS6において、a [ptr] および b [ptr] をレジスタファイル4にロードしなければならないが、これらのデータが、データキャッシュ8あるいはデータキャッシュ9にコピーされていて、キャッシュミスなしにロードされるという保証はない。従って、最悪の場合、キャッシュミスを起こして、ステップS6における計算処理に時間がかかりすぎることを考えられる。また、ロカブルなキャッシュ制御はプログラムに提供されてはいるが、これをどのように使えばキャッシュミスなしで高速に計算できるのかが知られていなかった。

【0050】このように、従来、V L Cを情報処理装置で計算させる場合、短い時間で処理しなければならないデータについて、キャッシュミスを起こし、データの処理に時間が長くなり過ぎてしまう場合がある課題があった。そのため、キャッシュミスを起こさずに確実に短い時間で処理したいという要望には答えられていなかった。

【0051】本発明はこのような状況に鑑みてなされたものであり、比較的短い時間内で処理しなければならないデータについては、キャッシュミスなしで高速に読み出せるようにし、その時間内で確実に処理を行うことができるようにするものである。

【0052】

【課題を解決するための手段】請求項1に記載のキャッシュメモリ制御方法は、データのうち、所定の時間内で参照すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたデータの所定のものをキャッシュメモリに常駐させることを特徴とする。

【0053】請求項2に記載のキャッシュメモリ制御方法は、プログラムのうち、所定の時間内で実行すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラムの所定のものをキャッシュメモリに常駐させることを特徴とする。

【0054】請求項3に記載の変長符号化方法は、第1のメモリに、入力データに対応する符号化された圧縮データを記憶させ、第1のメモリに記憶された圧縮データのうち、圧縮データの符号長が所定のビット数より小

さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された圧縮データをキャッシュメモリに常駐させることを特徴とする。

【0055】請求項4に記載の変長符号化方法は、入力データに対応する圧縮データを生成するプログラムを第2のメモリに記憶させ、第2のメモリに記憶されたプログラムのうち、プログラムが生成する圧縮データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラムをキャッシュメモリに常駐させることを特徴とする。

【0056】請求項5に記載の変長復号化方法は、第1のメモリに、圧縮された入力データに対応する復号化データを記憶させ、第1のメモリに記憶された復号化データのうち、対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された復号化データをキャッシュメモリに常駐させることを特徴とする。

【0057】請求項6に記載の変長復号化方法は、入力データに対応する復号化データを生成するプログラムを第2のメモリに記憶させ、第2のメモリに記憶されたプログラムのうち、プログラムが生成する復号化データに対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラムをキャッシュメモリに常駐させることを特徴とする。

【0058】請求項7に記載のキャッシュメモリ制御方法においては、データのうち、所定の時間内で参照すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたデータの所定のものをキャッシュメモリに常駐させる。従って、比較的、短時間で参照すべきデータを高速に参照することができる。

【0059】請求項8に記載のキャッシュメモリ制御方法においては、プログラムのうち、所定の時間内で実行すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラムの所定のものをキャッシュメモリに常駐させる。従って、比較的、短時間で実行すべきプログラムを迅速に実行させることができる。

【0060】請求項9に記載の変長符号化方法においては、第1のメモリに、入力データに対応する符号化された圧縮データを記憶させ、第1のメモリに記憶された圧縮データのうち、圧縮データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された圧縮データをキャッシュメモリに常

駐させる。従って、符号長の短い圧縮データに対して高速にアクセスすることができる。

【0061】請求項 4に記載の可変長符号化方法においては、入力データに対応する圧縮データを生成するプログラムを第2のメモリに記憶させ、第2のメモリに記憶されたプログラムのうち、プログラムが生成する圧縮データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラムをキャッシュメモリに常駐させる。従って、符号長の短い圧縮データを生成するプログラムを迅速に実行させることができる。

【0062】請求項 5に記載の可変長復号化方法においては、第1のメモリに、圧縮された入力データに対応する復号化データを記憶させ、第1のメモリに記憶された復号化データのうち、対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された復号化データをキャッシュメモリに常駐させる。従って、符号長の短い圧縮データに対応する復号化データに対して高速にアクセスすることができる。

【0063】請求項 6に記載の可変長復号化方法においては、入力データに対応する復号化データを生成するプログラムを第2のメモリに記憶させ、第2のメモリに記憶されたプログラムのうち、プログラムが生成する復号化データに対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラムをキャッシュメモリに常駐させる。従って、符号長の短い圧縮データに対応する復号化データを生成するプログラムを迅速に実行させることができる。

【0064】
【発明の実施の形態】図1は、本発明を適用した情報処理装置の一実施例の構成を示すブロック図である。図1に示した装置は、図13に示した従来の装置の場合と基本的に同様の構成をなしている。すなわち、図1において、制御回路（CONTROL）1は、装置全体を制御するようになされている。インストラクションメモリ（IMEM）2は、所定のインストラクション（プログラムのコード等）を記憶する。データメモリ（DMEM）7は、所定のデータを記憶する。レジスタファイル（REG）4は、制御回路1からの指令に基づいて、後述するセレクト、およびデータキャッシュ（DCA）1またはデータキャッシュ2より、データ用バス（DBUS）6を介して供給されたデータを記憶し、算術演算回路（ALU）5に供給するようになされている。データ用バス6は、レジスタファイル4と算術演算回路5との間で、データの受渡しを行うようになされて

いる。

【0065】レジスタファイル4、算術演算回路5、データ用バス6、および制御回路1の一部（点線で囲まれた部分）は、いわゆるCPU（Central Processing Unit）3を構成している。

【0066】セレクト10は、制御回路1によって制御され、データメモリ7からのデータをデータキャッシュ8またはデータキャッシュ9のいずれかに供給したり、データキャッシュ8またはデータキャッシュ9のデータをデータメモリ7に供給するようになされている。データキャッシュ8およびデータキャッシュ9は高速にデータの読み書きが可能なメモリであり、データキャッシュ8は、セレクト10を介してデータメモリ7からデータが供給されたとき、制御回路1の制御により、そのデータを記憶する。また、データキャッシュ9は、セレクト10を介してデータメモリ7からデータが供給されたとき、制御回路1の制御により、そのデータを記憶するようになされている。

【0067】セレクト11は、制御回路1の制御に基づいて、データキャッシュ8またはデータキャッシュ9のいずれかに記憶されたデータをデータ用バス6を介して、レジスタ4に供給したり、あるいは、データ用バス6を介して、レジスタファイル4からのデータをデータキャッシュ8またはデータキャッシュ9に供給するようになされている。

【0068】インストラクションメモリ2には、処理すべきプログラムの内容（コード）が格納されている。制御回路1からの制御により、インストラクションメモリ2内のプログラムのコードが順番に読み出され、読み出されたコードを制御回路1が解釈して、適切なデータをデータメモリ7から読み出し、レジスタファイル4と算術演算回路5を使用して所定の演算を行い、データメモリ7に演算結果を戻すようになされている。

【0069】図2は、図1の実施例において実行されるVLC（Variable Length Coding：可変長符号化）の全体のフローチャートを示している。これは、図14に示した従来の情報処理装置におけるVLCの全体のフローチャートと基本的に同様である。また、本発明においては、図1のステップS31において行われる初期設定のサブルーチンが、図3に示すサブルーチンとなり、従来の例である図15に示したものと異なっている。

【0070】図2に示したフローチャートのステップS32乃至S39は、図14に示したフローチャートのステップS2乃至S9にそれぞれ対応しており、各ステップにおける処理内容は、図14を参照して上述した場合と基本的に同様であるので、ここでは、ステップS31の処理について説明し、その他のステップについての説明は適宜省略する。

【0071】以下、図3のフローチャートを参照して、図1に示した情報処理装置において、初期設定が行われ

るときの動作について説明する。まず、ステップS41において、図15のステップS21の場合と同様に、図7乃至図10に示す符号を格納しておく領域a[0]乃至a[32×40-1]と、その符号がどれほどの長さ(ビット長)であるかを記憶しておくための領域b[0]乃至b[32×40-1]をデータメモリ7(図1)上に確保する。そして、図7乃至図10に示す各(run, levelの絶対値)に対応する符号を、a[(levelの絶対値-1)×32+run]に上位側につめて格納する。

【0072】また、図7乃至図10に示す各(run, levelの絶対値)に対応する符号の長さ(ビット長)をb[(levelの絶対値-1)×32+run]に格納する。対応する(run, levelの絶対値)が図7乃至図10にないために、まだデータが格納されていない残りのb[j]には24を格納する。但し、図7乃至図10に示す符号の最後のsについては無視する。

【0073】次に、ステップS42において、データキャッシュ8またはデータキャッシュ9のいずれか片方をロックする。例えば、この場合、データキャッシュ9をロックする。

【0074】次に、ステップS43において、a[0]、a[1]、a[32]、a[2]、a[54]、a[3]、a[4]、a[33]、a[5]、a[6]、a[7]、b[0]、b[1]、b[32]、b[2]、b[54]、b[3]、b[4]、b[33]、b[5]、b[6]、およびb[7](以下では、これらのデータのことを「a[0]、およびb[0]等のデータ」ということにする)をロードする。

【0075】ここではレジスタファイル4にロードするだけで、これらのデータを使用して演算を行うことはしない。これらのデータのロードによって、最初、上述したa[0]、およびb[0]等のデータは、データキャッシュ8およびデータキャッシュ9のいずれにもないものでキャッシュミスが起こり、データメモリ7からデータキャッシュ8またはデータキャッシュ9のいずれかにコピーされる。

【0076】これは、初期値の設定であり、実際のVLCを行う前段階であるから、キャッシュミスを起こして時間がいくらかかっても良い。このとき、ステップS42において、データキャッシュ9がロックされているので、データメモリ7からのデータのコピーは、データキャッシュ8に対して行われる。つまり、上述したa[0]、およびb[0]等のデータは全てデータキャッシュ8にコピーされる。

【0077】ここで、上述したa[0]、およびb[0]等のデータは、図7乃至図10に示す符号のうち、7ビット(sを除けば6ビット)以下のビット長を有する符号である。

【0078】次に、ステップS44において、データキャッシュ8をロックし、データキャッシュ9のロックを解除する。従って、これ以降は、キャッシュミスを起こした場合、常に、データキャッシュ9にデータがコピーされる。そして、データキャッシュ8にはオーバーライトされることはないので、上述したa[0]、およびb[0]等のデータはデータキャッシュ8上に常駐することになる。

【0079】そして、このサブルーチンの処理を終了し、リターンする。

【0080】さて、このように初期設定をすることで、例えば、図2のステップS33において、(run, level)=(3, 1)が読み込まれると、ステップS35において、ptr=(levelの絶対値-1)×32+run=(1-1)×32+3=3が計算され、ステップS36において、a[3]、およびb[3]が読み出され、これらがそれぞれcode tmp、およびcode lengthとされる。

【0081】上記処理を、実際に図1に示した情報処理装置を使って計算する場合、a[3]、およびb[3]はすでにデータキャッシュ8にコピーされているので、ステップS36において、キャッシュミスを起こすことなくそれらのデータa[3]およびb[3]を読み出すことができる。そして、ステップS38においては、(run, level)=(3, 1)の符号である"001110"が出力される。

【0082】この場合において、例えば5Mbit(Mビット)/sec(秒)の一定レートで上記圧縮したデータ(符号化されたデータ)を出力しなければならないとき、VLCは、6×200nsec(ナノ秒)以内にステップS33乃至S38の処理を行わなければならない。なぜなら、(run, level)=(3, 1)の時は、符号長は6ビットであるからである。しかしながら、ステップS36においては、キャッシュミスを起こさないで、上記時間内に計算することが可能である。

【0083】また、例えば、図2のステップS33において、(run, level)=(1, -2)が読み込まれると、ステップS35において、ptr=(2-1)×32+1=33が計算され、ステップS36においては、a[33]、およびb[33]が読み出され、これらがそれぞれcode tmp、およびcode lengthとされる。

【0084】上記処理を、実際に図1に示した情報処理装置を使って計算する場合、a[33]、およびb[33]はデータキャッシュ8にすでにコピーされているので、キャッシュミスすることなく読み出すことができる。そして、ステップS38においては、(run, level)=(1, -2)の符号である"0001101"が出力される。

【0085】この場合において、例えば5Mbit/sの一定レートで上記圧縮したデータ（符号化されたデータ）を出力しなければならないとき、VLCは、 $7 \times 200nsec$ 以内にステップS33乃至S38の処理を行わなければならない。なぜなら、 $(run, level) = (1, -2)$ のときは、図7から分かるように、符号長が7ビットであるからである。しかしながら、ステップS36において、キャッシュミスを起こさないで、上記時間内に計算することが可能である。ちなみに、従来の場合には、キャッシュミスを起こす可能性もあり、上記時間内に確実に計算できるとは言い切れなかった。

【0086】また、例えば、図2のステップS33において、 $(run, level) = (1, 18)$ が読み込まれると、ステップS35において、 $ptr = (18 - 1) \times 32 + 1 = 545$ が計算され、ステップS36においては、 $a[545]$ 、および $b[545]$ が読み出され、これらがそれぞれ $codetmp$ 、および $code length$ とされる。

【0087】しかしながら、上記処理を、実際に図1の装置を使って計算するとき、 $a[545]$ 、および $b[545]$ はデータキャッシュ8にはコピーされていない。そして、データキャッシュ9にもコピーされていない。従って、最悪の場合、キャッシュミスを起こして、データメモリ7からデータキャッシュ9へ、 $a[545]$ 、および $b[545]$ をコピーしなければならない。そのため、この間、計算が一時的に中断され、コピーを終了した時点で計算が再開される。

【0088】ステップS38においては、 $(run, level) = (1, 18)$ の符号である“000000000000100000”が出力される。この場合、上述したようにキャッシュミスを起こすので、ステップS36における処理に時間がかかり、ステップS33乃至S38の一連の処理には時間がかかることになる。しかしながら、 $(run, level) = (1, 18)$ のときは、符号長は17ビットであるから、 $17 \times 200 (= 3400)nsec$ 以内に、ステップS33乃至S38の処理を行えば良い。従って、キャッシュミスを起こしても、上記時間内に計算することが可能である。

【0089】このように、一定レートで圧縮したデータを出力するとき、符号長の比較的短いものに対しては比較的短い時間で計算しなければならないが、上述したように、符号長の短いものは予めデータキャッシュ8にロードし、ロックしているので、キャッシュミスを起こすことなく読み出すことができる。従って、ステップS33乃至S38の処理（計算）を制限時間内に実行することができる。

【0090】一方、符号長の比較的長いものに対しては比較的長い時間をかけてステップS33乃至S39の処理（計算）を行っても良いので、キャッシュミスを起こ

しながら計算しても間に合う。このように、本発明を適用することにより、符号長が短い場合、および符号長が長い場合において、それぞれの場合で処理しなければならない時間に適した時間内で計算することができる。

【0091】また、上述の例では、データメモリ7上に、ルックアップテーブルとしてのデータ $a[i]$ 、および $b[i]$ （ここで、 $i = (levelの絶対値 - 1) \times 32 + run$ ）を持ち、所定の計算を行うようにしたが、以下のようにして計算することも可能である。即ち、 $(run, level) = (3, 1)$ のときはサブルーチン $(sub3 - 1)$ を読み出すようにし、 $(run, level) = (1, -2)$ のときはサブルーチン $(sub1 - 2)$ を読み出すようにし、 $(run, level) = (1, 18)$ のときはサブルーチン $(sub1 - 18)$ を読み出すようにする。

【0092】ここで、サブルーチン $sub3 - 1$ は“00111s”を出力させる処理を実行するサブルーチンであり、サブルーチン $sub1 - 2$ は“000110s”を出力させる処理を実行するサブルーチンであり、サブルーチン $sub1 - 18$ は“0000000000010000s”を出力させる処理を実行するサブルーチンである。sは、levelの符号（正、負）である。

【0093】このように、図7乃至図10に示す各 $(run, levelの絶対値)$ に対して、対応する符号を出力するサブルーチン $sub[run] - [levelの絶対値]$ をあらかじめ用意しておき、読み込んだ $(run, level)$ に対して、サブルーチン $sub[run] - [levelの絶対値]$ を読み出すことにより、VLCを行うようにすることもできる。

【0094】本発明は、このようなサブルーチンを使った場合にも適用することができる。図4は、本発明を適用した情報処理装置の他の実施例の構成を示すブロック図である。図4においては、図1において、インストラクション（この場合サブルーチン）を記憶しておくための、インストラクションキャッシュ22およびインストラクションキャッシュ23を新たに設けるようにしている。従って、図4に示した情報処理装置においては、データメモリ7内のデータを高速に読み出すことができるとともに、インストラクションメモリ（IMEM）2内のプログラム（サブルーチン）を、制御回路1に制御されたセクタ24を介して高速に読み出すことができるようになされている。

【0095】また、制御回路1に制御されたセクタ21を介して、インストラクションキャッシュ22またはインストラクションキャッシュ23内に記憶されたプログラム（サブルーチン）が制御回路1に供給されるようになされている。制御回路1は、このサブルーチンに基づいて、所定の処理を実行することになる。

【0096】上述のサブルーチンの内、例えば、7ビッ

ト以下の符号を出力させるためのサブルーチンをあらかじめインストラクションキャッシュ22またはインストラクションキャッシュ23にコピーさせ、ロックしておく。具体的には、サブルーチン(sub) 0-1, sub1-1, sub0-2, sub2-1, sub0-3, sub3-1, sub4-1, sub1-2, sub5-1, sub6-1, およびsub7-1の11個のサブルーチンをインストラクションキャッシュ22または23にコピーし、ロックする。

【0097】これにより、これらのサブルーチンはインストラクションキャッシュ22またはインストラクションキャッシュ23上に常駐しているため、キャッシュミスなしで読み出すことが可能である。つまり、高速に読み出され、制御回路1により、そのサブルーチンに従った処理が実行される。上記以外のサブルーチンは読み出されるときキャッシュミスを起こす可能性があるが、符号長が8ビット以上と長いので、キャッシュミスを起こしながら処理(計算)させても間に合う。

【0098】以上のように、本発明は、VLCの計算において、出力する符号長は可変であり、また、出力する圧縮されたデータのレートはMPEG2 (Moving Picture Experts Group phase 2) では一定(例えば5Mbit (メガビット) / sec (秒))であることに着目したものである。すなわち、符号長の短い符号をVLCの計算を行って出力する場合、この計算を高速に実行する必要があり、符号長の長い符号をVLCの計算を行って出力する場合、この計算は比較的ゆっくり実行してもかまわない。従って、上記実施例においては、符号長に応じた適切な時間内で符号化を行うようにしている。

【0099】また、上述の例では、VLCについて述べたが、VLD (Variable Length Decoding: バリエーブルレングスデコーディング) についても適用することが可能である。VLDとは、圧縮されたデータ(符号化されたデータ)を元に復元する操作である。具体的には、図7乃至図10に示す符号から、(run, level)を生成する操作を行うものである。

【0100】VLDは、例えば、以下のようにして計算される。即ち、入力される圧縮されたデータ(符号化されたデータ)が、"00111s"であれば、サブルーチン(subsub3-1)を読み出すようにし、"000110s"であれば、サブルーチン(subsub1-2)を読み出すようにし、"00000000000010000s"であれば、サブルーチン(subsub1-18)を読み出すようにする。

【0101】ここで、サブルーチンsubsub3-1は、run=3とし、level=1 (この場合、s=0)、またはlevel=-1 (この場合、s=1)とするサブルーチンであり、サブルーチンsubsub1-2は、run=1とし、level=2 (この場合、s=0)、またはlevel=-2 (この場合、s=

1)とするサブルーチンであり、サブルーチンsubsub1-18は、run=1とし、level=18 (この場合、s=0)、またはlevel=-18 (この場合、s=1)とするサブルーチンである。

【0102】このように、図7乃至図10に示す各符号に対して、対応する(run, level)を出力するサブルーチンsubsub[run]-[level]の絶対値をあらかじめ用意しておき、読み込んだ符号に対して、サブルーチンsubsub[run]-[level]の絶対値を読み出すことで、VLDを行うことができる。

【0103】本発明は、このようなVLDにも適用可能である。上述したサブルーチンの内、例えば入力される符号が7ビット以下のものであるサブルーチンをあらかじめインストラクションキャッシュ22または23のいずれかにコピーさせ、ロックしておく。具体的には、subsub0-1, subsub1-1, subsub0-2, subsub2-1, subsub0-3, subsub3-1, subsub4-1, subsub1-2, subsub5-1, subsub6-1, およびsubsub7-1の11個のサブルーチンをインストラクションキャッシュ22または23のいずれかにコピーし、ロックする。

【0104】例えば5Mbit/secの一定レートで圧縮したデータ(符号化されたデータ)が入力されてくる場合、符号長が8ビット以上の比較的長いものは、8×200 (=1600) nsec以上かけて上記サブルーチンを読み出して処理を行えば良いので、キャッシュミスを起こしても間に合う。もし、高速に処理したとしても、一定レートで入力されてくるので、次の符号が入力されず単に待つだけである。また、符号長が7ビット以下の比較的短いものは、速く上記サブルーチンを読み出して処理を行わなければならないが、それに対応するサブルーチンはインストラクションキャッシュ22または23のいずれかに常駐しているため、キャッシュミスなしで読み出すことができ、高速に処理することが可能である。

【0105】また、VLDを行う場合において、上述したように、サブルーチンを用いるのではなく、VLCの場合と基本的に同様にして、例えば7ビット以下の符号長を有する符号に対応するデータ(run, level)を、データキャッシュ8またはデータキャッシュ9のいずれかに記憶させ、ロックする。そして、これに基づいて、VLDを行わせるようにすることも可能である。

【0106】なお、上記実施例においては、本発明をMPEG2方式による符号化または復号化に適用する場合について説明したが、本発明は、これに限定されるものではない。

【0107】また、上記実施例は、2つのデータキャッ

シユおよび2つのインストラクションキャッシュを備えた構成としたが、上記構成に限定されるものではない。

【0108】また、上記実施例においては、データキャッシュに常駐させるデータを符号長が7ビット以下の符号としたが、これに限定されるものではない。同様に、インストラクションキャッシュに常駐させるデータを符号長が7ビット以下の符号に対応するサブルーチンとしたが、これに限定されるものではない。

【0109】また、上記実施例において、符号化または復号化の対象とするデータは、画像データに限らず、音声データやその他のデータとすることができる。

【0110】

【発明の効果】請求項1に記載のキャッシュメモリ制御方法によれば、データのうち、所定の時間内で参照すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたデータの所定のものをキャッシュメモリに常駐させるようにしたので、比較的、短時間で参照すべきデータを高速に参照することができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのデータを参照することが可能となる。これにより、データに応じた適切な時間で処理を行うことができる。

【0111】請求項2に記載のキャッシュメモリ制御方法によれば、プログラムのうち、所定の時間内で実行すべきものを予めキャッシュメモリに転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラムの所定のものをキャッシュメモリに常駐させるようにしたので、比較的、短時間で実行すべきプログラムを迅速に実行させることができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのプログラムを実行させることが可能となる。これにより、データに応じた適切な時間で処理を行うことができる。

【0112】請求項3に記載の可変長符号化方法によれば、第1のメモリに、入力データに対応する符号化された圧縮データを記憶させ、第1のメモリに記憶された圧縮データのうち、圧縮データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された圧縮データをキャッシュメモリに常駐させるようにしたので、符号長の短い圧縮データに対して高速にアクセスすることができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのデータを読み出すことが可能となる。これにより、符号長に応じた適切な時間で符号化処理を実行することができる。

【0113】請求項4に記載の可変長符号化方法によれば、入力データに対応する圧縮データを生成するプログラムを第2のメモリに記憶させ、第2のメモリに記憶されたプログラムのうち、プログラムが生成する圧縮データ

の符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラムをキャッシュメモリに常駐させるようにしたので、符号長の短い圧縮データを生成するプログラムを迅速に実行させることができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのプログラムを実行させることが可能となる。これにより、符号長に応じた適切な時間で符号化処理を実行することができる。

【0114】請求項5に記載の可変長復号化方法によれば、第1のメモリに、圧縮された入力データに対応する復号化データを記憶させ、第1のメモリに記憶された復号化データのうち、対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送された復号化データをキャッシュメモリに常駐させるようにしたので、符号長の短い圧縮データに対応する復号化データに対して高速にアクセスすることができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのデータを読み出すことが可能となる。これにより、符号長に応じた適切な時間で復号化処理を実行することができる。

【0115】請求項6に記載の可変長復号化方法によれば、入力データに対応する復号化データを生成するプログラムを第2のメモリに記憶させ、第2のメモリに記憶されたプログラムのうち、プログラムが生成する復号化データに対応する入力データの符号長が所定のビット数より小さいものを、キャッシュメモリに予め転送し、キャッシュメモリをロックすることにより、キャッシュメモリに転送されたプログラムをキャッシュメモリに常駐させるようにしたので、符号長の短い圧縮データに対応する復号化データを生成するプログラムを迅速に実行させることができ、キャッシュメモリをロックするので、キャッシュミスを起こすことなく、そのプログラムを実行させることが可能となる。これにより、符号長に応じた適切な時間で復号化処理を実行することができる。

【図面の簡単な説明】

【図1】本発明を適用した情報処理装置の一実施例の構成を示すブロック図である。

【図2】図1に示した実施例の動作を説明するためのフローチャートである。

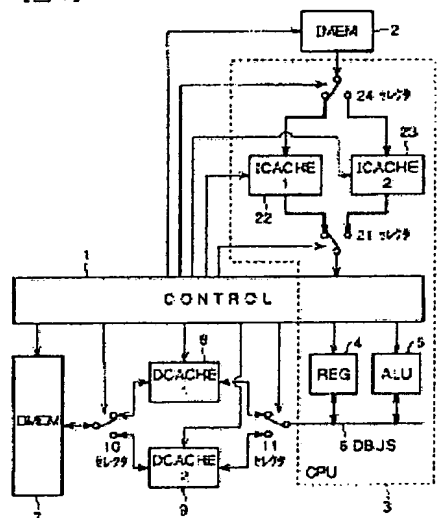
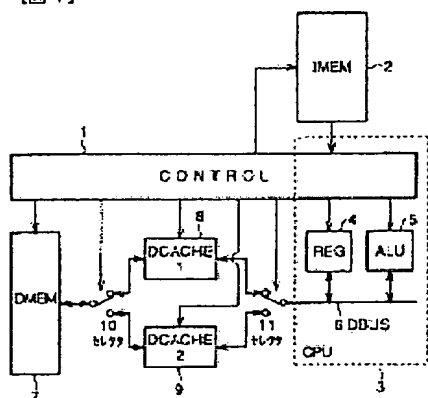
【図3】図2のフローチャートのステップS31に対応するサブルーチンの処理を説明するためのフローチャートである。

【図4】本発明を適用した情報処理装置の他の実施例の構成を示すブロック図である。

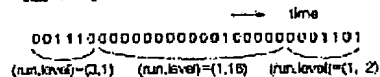
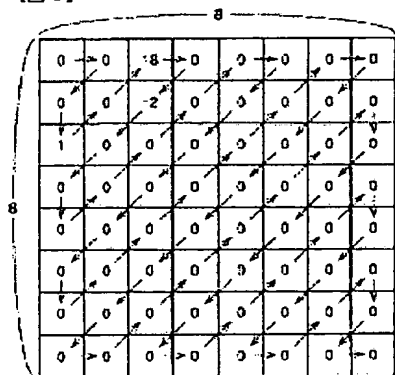
【図5】ジグザグスキャンデータ列とラインスキャンデータ列を示す図である。

【図6】図5に対応するデータ(run, level)

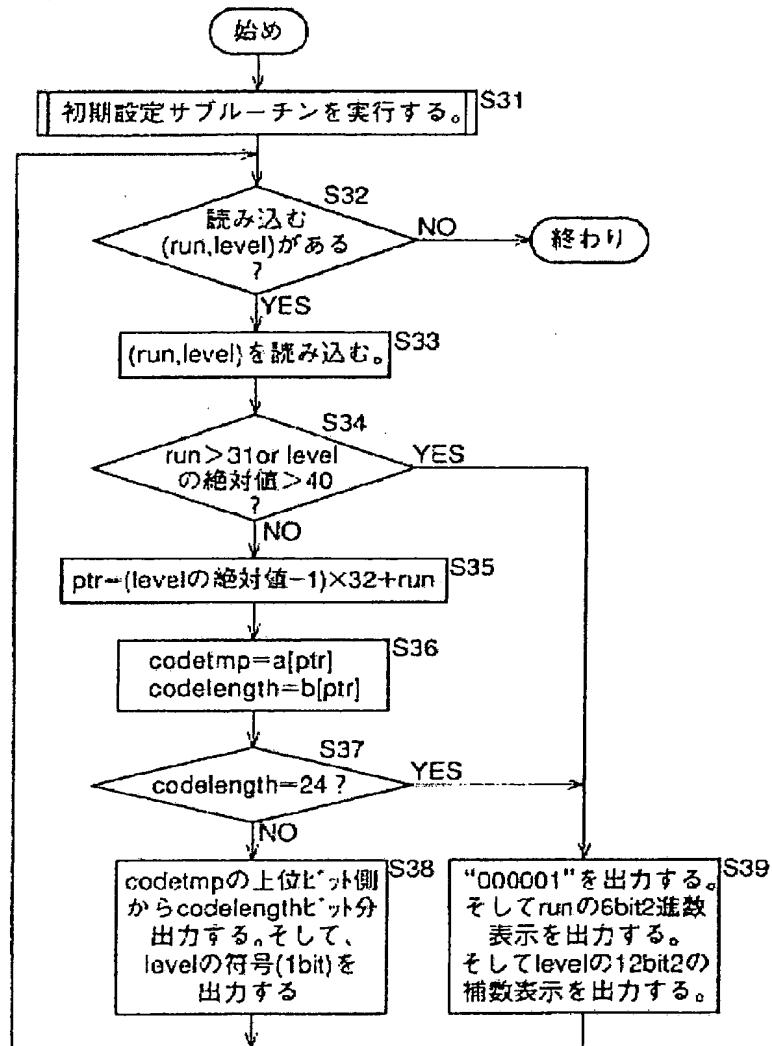
【図 14】 図 13 に示した装置の動作を説明するための



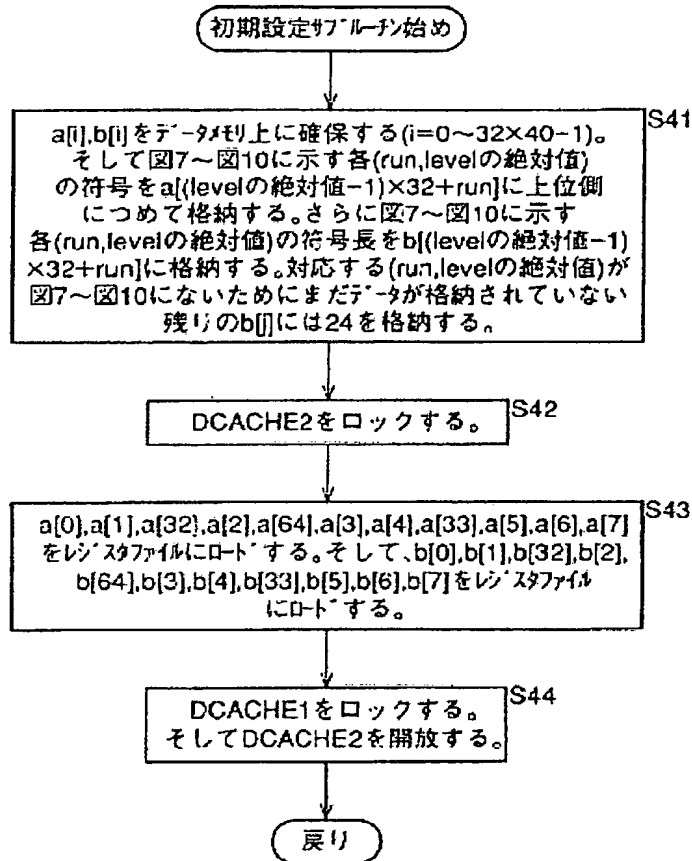
run	3	1	1
love!	1	18	-2



【図2】



【図3】



【図12】

Table 6-18 ... Encoding of run and level following an ESCAPE code

fixed length code	run	fixed length code	signed level
0000 00	0	1000 0000 0001	-2047
0000 01	1	1000 0000 00 0	-2046
0000 10	2
...	...	1111 1111 1111	-1
...	...	0000 0000 0000	forbidden
...	...	0000 0000 0001	+1
...
1111 11	63	0111 1111 1111	+2047

【図 7】

Table B-14 --- DCT coefficients Table zero

Variable length code (NOTE1)	run	levelの絶対値 (levelの絶対値 -1) X 32 + run
11s	0	1
011s	1	1
0100s	0	2
0101s	2	1
00101s	0	3
00111s	3	1
00110s	4	1
000110s	1	2
000111s	5	1
000101s	6	1
000100s	7	1
0000110s	0	4
0000100s	2	2
0000111s	8	1
0000101s	9	1
000001	Escape	
00100110s	0	5
00100001s	0	0
00100101s	1	3
00100100s	3	2
00100111s	10	1
00100011s	11	1
00100010s	12	1
00100000s	13	1
0000001010s	0	7
0000001100s	1	4
0000001011s	2	3
0000001111s	4	2
0000001011s	5	2
0000001110s	14	1
0000001101s	15	1
0000001010s	16	1

NOTE1: The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.

【図 8】

Table B-14 --- DCT coefficients Table zero (continued)

Variable length code (NOTE1)	run	levelの絶対値
000000011101s	10	8
000000011000s	10	9
000000010011s	10	10
000000010000s	0	11
000000010011s	1	5
000000010100s	2	4
000000011100s	3	3
000000010010s	4	3
000000011110s	8	2
000000010101s	7	2
000000010001s	8	2
000000011111s	17	1
000000010101s	18	1
000000010001s	19	1
000000010111s	20	1
000000010110s	21	1
000000011010s	0	12
000000011001s	0	13
000000011000s	0	14
000000010111s	0	15
000000010110s	1	8
000000010101s	1	7
000000010100s	2	5
000000010011s	3	4
000000010010s	5	3
000000010001s	9	2
000000010000s	10	2
000000011111s	22	1
000000011110s	23	1
000000011101s	24	1
000000011100s	25	1
000000011011s	26	1

NOTE1: The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.

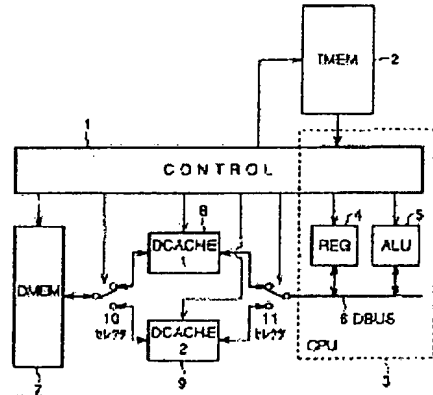
【図 10】

Table B-14 --- DCT coefficients Table zero (concluded)

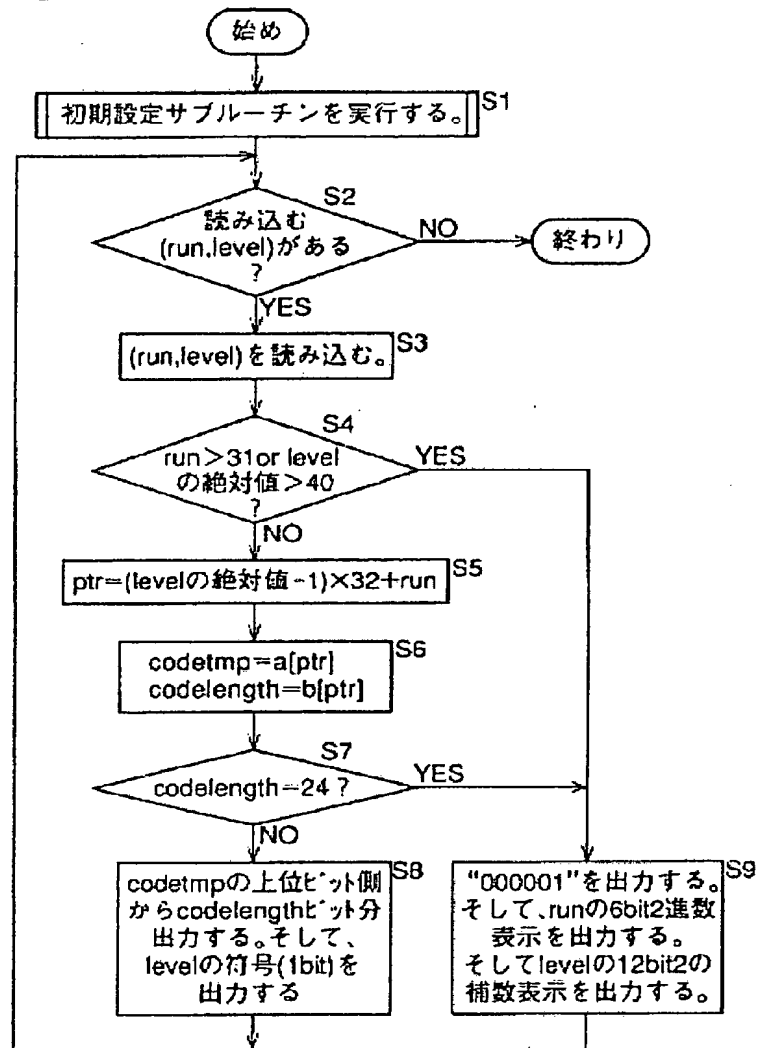
Variable length code (NOTE1)	run	levelの絶対値
000000000010111s	1	15
000000000010101s	1	16
000000000010001s	1	17
000000000010000s	1	18
000000000010100s	6	3
000000000010101s	11	2
000000000010001s	12	2
000000000010000s	13	2
000000000010111s	14	2
000000000010110s	15	2
000000000010101s	16	2
000000000011111s	27	1
000000000011110s	28	1
000000000011101s	29	1
000000000011100s	30	1
000000000010111s	31	1

NOTE1: The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.

【図 13】



【図14】



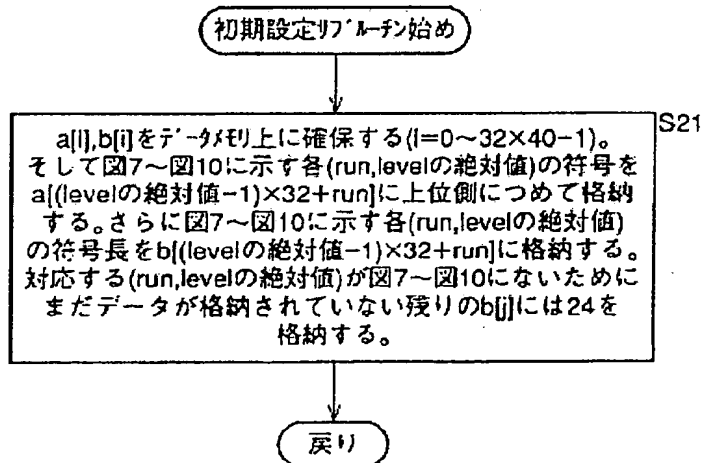
【図9】

Table B-14 ... DCT coefficients Table zero(continued)

Variable length code(NOTE):run	levelの絶対値
0000 0000 0111 11 s	16
0000 0000 0111 10 s	17
0000 0000 0111 01 s	18
0000 0000 0111 00 s	19
0000 0000 0110 11 s	20
0000 0000 0110 10 s	21
0000 0000 0110 01 s	22
0000 0000 0110 00 s	23
0000 0000 0101 11 s	24
0000 0000 0101 10 s	25
0000 0000 0101 01 s	26
0000 0000 0101 00 s	27
0000 0000 0100 11 s	28
0000 0000 0100 10 s	29
0000 0000 0100 01 s	30
0000 0000 0100 00 s	31
0000 0000 0011 000 s	32
0000 0000 0010 111 s	33
0000 0000 0010 110 s	34
0000 0000 0010 101 s	35
0000 0000 0010 100 s	36
0000 0000 0010 011 s	37
0000 0000 0010 010 s	38
0000 0000 0010 001 s	39
0000 0000 0010 000 s	40
0000 0000 0011 111 s	8
0000 0000 0011 110 s	9
0000 0000 0011 101 s	10
0000 0000 0011 100 s	11
0000 0000 0011 011 s	12
0000 0000 0011 010 s	13
0000 0000 0011 001 s	14

NOTE: The symbol 's' denotes the sign of the level, '1' for positive, '0' for negative.

【図15】



**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☒ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.